REMARKS

Claims 1-3 and 5-34 are pending. Claims 1, 7, 17, 21, and 31 are independent. Claims 1, 7, 17, 21, and 31 are amended without adding new matter to recite that the instance of the business process specifies error compensation. See, e.g., Published Application, ¶ 0063. Claims 1-3 and 5-34 are rejected under 35 U.S.C. § 103. Reconsideration of the rejection in view of the following remarks is respectfully requested.

Telephone Conversation With Examiner

Examiner Anya is thanked for the telephone conversation conducted on April 8, 2009. Proposed claim amendments were discussed. Cited art was discussed. Although no agreements were reached, it appears that the proposed amendments may overcome the rejections based on the cited art.

Rejection of Claims 1-3 and 5-34 under 35 U.S.C. § 103(a)

Claims 1-3 and 5-34 are rejected under 35 U.S.C. § 103(a) as being unpatentable over (1) U.S. Patent Application Publication No. 2003/0093500, by Khodabakchian *et al.* (hereinafter referred to as "Khodabakchian") in view of at least one of: (2) U.S. Patent Application Publication No. 2005/0086297, by Hinks (hereinafter referred to as "Hinks"); (3) U.S. Patent Application Publication No. 2005/0027559, by Rajan *et al.* (hereinafter referred to as "Rajan") and U.S. Patent Application Publication No. 2002/0111996, by Jones *et al.* (hereinafter referred to as "Jones"). Specifically, claims 1, 2, 7, 17, 21 and 31 are rejected in view of Khodabakchian and Hinks, claims 1, 2, 5-10, 12-24 and 26-34 are rejected in view of Khodabakchian and Rajan, and claims 3, 11 and 25 are rejected in view of Khodabakchian, Rajan and Jones. (Office Action, pp. 2-14.) Applicants respectfully traverse the rejections.

It is respectfully submitted that the combination of references fails to teach or suggest the claimed subject matter. In summary, the Office Action admits Khodabakchian fails to teach or suggest the claimed subject matter because it teaches a centralized exception handler 202 (FIG. 2), but alleges that Hinks and Rajan teach an instance having exception handling code or failure

handling functionality. Applicants point out in detail below that Hinks and Rajan, like Khodabakchian, fail to teach an instance having exception handling code or failure handling functionality. All three references teach only centralized error handlers separate from instances of processes. An additional patentable distinction has been added by amendment, reciting that the exception handling code or failure handling functionality specifies error compensation.

Khodabakchian discusses a system "handling processes that interact with one or more asynchronous systems." Khodabakchian at ¶ 0005. Khodabakchian states that, "Web service orchestration server 102 contains multiple scenarios 110(1), 110(2) and 110(3)." Khodabakchian at ¶ 0023. "A scenario is a programming abstraction of a long-running process or a collaborative process." Id. "An exception handler 202 processes exceptions that are generated by orchestration engine 200 when implementing the instructions and commands in a scenario 110. Exceptions may include, for example, a fault generated by a web service or a notice generated as a result of a web service timeout." Khodabakchian at ¶ 0025; Figure 2. Khodabakchian indicates that failures are not processed within an instance of a process, but instead are processed by an independent "exception handler" 202 on web service orchestration server 102. Id.

The Office Action admits Khodabakchian fails to teach or suggest the claimed subject matter. (Office Action, p. 3). Contrary to the admission in the Office Action, Khodabakchian is not merely "silent" on the subject of exception handling code in an instance. Khodabakchian clearly indicates that exception handling in a separate centralized process 202 (FIG. 2) that is not present in scenarios.

The Office Action alleges that Hinks and Rajan teach what Khodabakchian fails to teach. However, it is respectfully submitted, Hinks and Rajan teach the same thing Khodabakchian teaches, which is a separate centralized error handler process shown in prior art FIG. 3 in the pending application.

Hinks. It is respectfully submitted that the Office Action's interpretation of Hinks (that it teaches instances "handle exceptions using the exception handling code within the instance") is

erroneous. Hinks discloses nothing but the prior art shown in FIG. 3 where error handling is a centralized service provided by the business framework separate from the business process. Compare the engine 300 in which main process 310 runs as shown in FIG. 3 of the pending application to Hinks where scripts run within Business Process Engine (BPE) 118. In Hinks, BPE 118 either provides a separate centralized error handler or, if not, it sends errors back to client 102, 104 for error handling.

The portion of Hinks being interpreted is its BPE 118, which is used by client 102 to orchestrate business processes in services network 106. Hinks, ¶ 0028. "BPE (118) provides extensions that facilitate the orchestration of the business process within the integration services network (106)." Hinks, ¶ 0029. In Hinks, BFW 112 centrally handles errors for business methods. Hinks, ¶ 0044 ("The BPE (118) provides error handling capabilities" unless no handler exists, in which case errors are returned to client 102 for handling).

The Office Action cites paragraph 0096, but it fails to teach or suggest the claim limitation it is asserted against. In accordance with the centralized error handling service provided by the BPE 118 or the return of an error to client 102 for error handling, paragraphs 0096-0098 of Hinks say:

[0096] . . . the only way the client (102) sending out the message (112) wants to be alerted to any failures and be given the opportunity to re-send the message (112) is by email. This is achieved by having the catch block of the multicast message (112) send an email that contains a link to the web-based messaging screen of the initiating service. The web-based messaging page generates a notification, containing the name of the service whose delivery failed, that is sent to the BPE (118). The BPE (118) handles the notification as an asynchronous event and resends the original request to the service that failed delivery. FIG. 4 shows a flowchart of the multicast process (400). . . ."

[0097] As can be seen in FIG. 4, the process (400) starts with the BPE (118) receiving a multicast script from the client (102) (step 402). The multicast script will be discussed in detail below. Next, the BPE (118) receives a message (112) from the client (102) with data, such as an email message to be sent to multiple recipients, and a reference to the BPEL multicast script (step 404). The BPE (118) then invokes the multicast process with the received data and sends the message to the specified recipients (step 406).

[0098] The process then checks whether the messages have been received by all the intended recipients (step 408). If the messages have been properly received, the process proceeds to step 416, where the result of the multicast process, such as a combined response from all the recipients or a confirmation, is sent back to the originating client (102) (step 416), upon which the process ends. If the any of the recipients have not received a copy of the message from the BPE (118), the BPE (118) notifies the client (102) about which recipients did not receive the message by sending an email containing a retry link to the client (102) (step 410).

Thus, Hinks fails to teach or suggest the claim limitation it is asserted against because Hinks only teaches centralized error handling by BPE 118 or the return of errors to client 102 for error handling.

Rajan. It is respectfully submitted that the Office Action's interpretation of Rajan (that it teaches instances "handle exceptions using the exception handling code within the instance") is clearly erroneous. Rajan discloses nothing but the prior art shown in FIG. 3 where error handling is a centralized service provided by the business framework separate from the business process. Compare the engine 300 in which main process 310 runs as shown in FIG. 3 of the pending application to Rajan where business methods run within Business Framework (BFW) 112. The portion of Rajan being interpreted is its business framework, which is "used to operate the [stateless] business objects themselves." Rajan, Abstract. In Rajan, BFW 112 centrally handles errors for business methods. Rajan wraps each invoked method with a "special try-catch block" to forward all errors to the centralized error handler. Wrapping all invoked methods clearly indicates error handling is not in the business methods. Wrapping means that each invoked method is assigned a proxy. Rajan, ¶ 0131. If an error occurs, the proxy forwards it to the centralized error handler in the BFW 112 and the invoked method has nothing to do with handling it. In detail, Rajan describes its system as follows:

"The business framework (BFW) 112 provides services for business objects. The business objects themselves run within the BFW 112." Rajan, ¶ 0074. "[A]ll business objects must be stateless. Because they are stateless, making multiple round-trips to do one task is impossible." Rajan, ¶ 0078. "The stateless business object, in their layer of generated code, can include methods for communicating with stored procedures." Rajan, ¶ 0031. "The set of central services on business objects includes messaging services . . . that enable asynchronous method invocation between business objects, and enable business objects to be invoked immediately or at a specific date and time." Rajan, ¶ 0035. "The scheduling services allow business objects to be invoked

once at a given date and time, or periodically at specified intervals." Rajan, ¶ 0037. "Certain business objects need to perform tasks at regular intervals. The BFW 112 preferably includes a scheduler component 1326. When the scheduling component 1326 is started, it 'looks' in a special table within the database called the schedule table. The scheduling table contains the name of the business object to invoke to perform the task, and the frequency of invocation." Rajan, ¶ 0129. "Communication into the present invention is always directed, initially, to the proxies. The form of communication is preferably by direct method invocation. . . . For instance, if the remote call is in the form of HTTP, then the remote call will be processed initially by a web server. That web server then translates the call to direct method invocation and calls the proxies." Rajan, ¶ 0089. "The BFW 112 has an advanced error-handling system." Rajan, ¶ 0094. "The error-handling system itself is easy to support. All errors flow through the same routine." Rajan, ¶ 0095. "Just as the BFW 112 uses try/catch blocks in each method, so does the CFW 106. This ensures that all errors flow through the error-handling routine of the present invention." Rajan, ¶ 0157. "The set of central services on business objects further includes error-handling services. The error-handling services support the capturing of operating system exceptions or communication applications (e.g., COM/COM+) errors." Rajan, ¶ 0024. "The present invention's error-handling feature is capable of capturing operating system errors by wrapping every method invocation in a special try/catch block. . . . In other words, in the preferred embodiment of the present invention, all errors flow through one error-handling routine." Rajan, ¶ 0096. "[I]f a developer happens to forget to wrap his method in a try/catch block, a compiler-error will be generated. Consequently, developers are encouraged to include try/catch blocks in every method, thus ensuring that all errors are handled by the error-handling system." Rajan, ¶ 0097. "The BFW 112 includes two key technologies that ensure reliability: structured exception handling; and clustering. As mentioned above, all of the method calls are typically wrapped in try/catch blocks. In fact, the compiler ensures that this is always the case. Structured exception handling is enforced by means of wrapping `try` and `catch` macros around the method invocations. A macro is a set of pre-defined instructions that are inserted at runtime. Structured exception handling causes all operating system failures to be passed to an error-handling routine." Rajan, ¶ 0099. "The BFW 112 also enables asynchronous method invocation via the asynchronous method invocation mechanism 1330. . . . A developer can elect this feature for a business object(s) by checking a checkbox in the database wizard 121. When the checkbox is set, an asynchronous proxy will be created which "wraps" the business object. The developer then invokes the asynchronous proxy instead of the business object. The asynchronous proxy will, in turn, invoke the business object (asynchronously)." Rajan, ¶ 0131.

As discussed by Rajan, wrapping creates a proxy. It does not modify the business object that is wrapped. One way or another, Rajan funnels all errors to the centralized error handler in the framework BFW 112. The business methods clearly do not handle errors. The fact that invoked business methods must be wrapped also indicates they cannot compensate for errors because they are without error handling code or failure handling functionality. If it were within

objects then there's no need for wrapping them to funnel errors to a centralized error handler. Thus, wrapping Khodabakchian's scenarios to forward errors to a centralized error handler as in Khodabakchian still completely fails to teach or suggest the claimed subject matter, as previously or presently presented.

Further, as previously noted, Khodabakchian fails to disclose "wherein the processing code has failure handling functionality" or "handles exceptions using the exception handling code within the instance." Application, Claims 1, 7, 17, 21 and 31. Also, the Office Action admits that Khodabakchian fails to disclose storing the instance after a predetermined time, but alleges Jones does. (Office Action, p. 12). It is respectfully submitted that this argument is inaccurate.

Jones discusses a host for executing an operation requested by requestor. Jones, ¶ 0018. The host requires a periodic signal from the requestor in order to process the request. Id. at ¶ 0020. If the periodic signal is not sent within a certain time then the host stores the request until a periodic signal is sent. Id.

Thus, Jones discloses the host storing the request if a periodic signal is not sent, not the requestor storing, after a predetermined time, the process that made the request while it waits for a response from the host. Thus, Jones does not provide the disclosure missing in Khodabakchian. Therefore, Applicants respectfully request withdrawal of the rejection.

Further still, each independent claim ahs been amended to recite that exception handling code or failure handling functionality in the instance specifies error compensation. This adds yet another distinction over the cited references.

The foregoing remarks apply in whole or in part to each pending claim and, therefore, it is respectfully submitted that the claimed subject matter, as previously and presently presented, is allowable over the cited references. Accordingly, Applicants respectfully request reconsideration

and withdrawal of the rejection of claims 1-3 and 5-34 under 35 U.S.C. § 103(a) in view of various combinations of Khodabakchian, Hinks, Rajan and Jones.

Amendments, made herein or previously made, are without abandonment of subject matter. Applicant expressly reserves the right to, in the pending application or any application related thereto, reintroduce any subject matter removed from the scope of claims by any amendment and introduce any subject matter not present in current or previous claims.

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that this application is in condition for allowance. Reconsideration of this application and an early Notice of Allowance are respectfully requested.

Date: June 5, 2009

/Joseph F. Oriti/
Joseph F. Oriti
Registration No. 47,835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439